НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 121 Інженерія програмного забезпечення

на тему Апаратно-програмний комплекс тестування технічного стану апаратних платформ

Виконав: студент 4 курсу, групи ТІ-61

Кулініч Ігор Олегович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Ковальчук А. М. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент доцент кафедри ТЕУ Т Та АЕС Сірий О.А _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення веб-технологій та мобільних пристроїв

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Кулініч Ігор Олегович
(прізвище, ім’я, по батькові)

1. Тема роботи Апаратно-програмний комплекс тестування технічного стану апаратних платформ
керівник роботи доцент, к.т.н. Ковальчук А. М.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 202__р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи: персональний комп’ютер, мови програмування C++, Java.
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні та апаратні рішення, можливі методи реалізації взаємодії елементів, обґрунтувати обрані програмні застосунки та шляхи розробки програмного забезпечення, розробити необхідне програмне та апаратне забезпечення, розробити інтерфейс користувача, зробити висновки за результатами роботи.
5. Перелік ілюстративного матеріалу
1. Моделювання системи 2. Апаратна архітектура системи. 3.Програмна архітектура системи. 4. Поєднання апаратних засобів системи. 5. Демонстрація роботи програмних та апаратних засобів системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”_2_” __грудня_____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	2.12.2019	
2.	Вивчення та аналіз задачі	2.12.2019-13.04.2020	
3.	Розробка архітектури та загальної структури системи	13.04.2020-27.04.2020	
4.	Розробка структур окремих підсистем	27.04.2020-04.05.2020	
5.	Програмна реалізація системи	20.04.2020-13.05.2020	
6.	Оформлення пояснювальної записки	04.05.2020-05.06.2020	
7.	Захист програмного продукту	17.05.2020	
8.	Передзахист	12.06.2020	
9.	Захист	15.06.2020	

Студент _____ _____Кулініч І.О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Ковальчук А.М._____
(підпис) (прізвище та ініціали,)

АННОТАЦИЯ

Цель работы - разработка аппаратно-программного комплекса тестирования технического состояния аппаратных платформ. В создании аппаратного комплекса были использованы Orange Pi, Arduino Mega и каскад мультиплексоров hc4067. Для разработки программного обеспечения были использованы язык C ++ (для контроллеров), веб-инструменты Java и фреймворк Vaadin. Разработан программно-аппаратный комплекс обеспечивает тестирование аппаратных платформ.

Записка содержит 73 страниц, 13 рисунков и 3 приложения.

АНОТАЦІЯ

Мета роботи - розробка апаратно-програмного комплексу тестування технічного стану апаратних платформ. У створенні апаратного комплексу були використані Orange Pi, Arduino Mega та каскад мультиплексорів hc4067. Для розробки програмного забезпечення було використано мову C++ (для контролерів), веб-інструменти Java та фреймворк Vaadin. Розроблений програмно-апаратний комплекс забезпечує тестування апаратних платформ.

Записка містить 73 сторінок, 13 рисунків та 3 додатки.

ABSTRACT

The purpose of the work is to develop a hardware-software complex for testing the technical condition of hardware platforms. Orange Pi, Arduino Mega and cascade of hc4067 multiplexers were used in the creation of the hardware complex. C++ (for controllers), Java web tools and the Vaadin framework were used to develop the software.

The developed hardware-software complex provides hardware platforms` testing.

Note note 73 pages, 13 figures and 3 documents.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

TCP/IP – Transmission Control Protocol/Internet Protocol

UART – Universal Asynchronous Receiver/Transmitter

HAL – Hardware Annotation Library

ACL – Agent Communication Language

KQML – Knowledge Query and Manipulation Language

MAS – Multi-agent systems

DMA – Direct Memory Access

CAN – Controller Area Network

RTC – Real-Time Clock

PWM – Pulse-Width Modulation

ВСТУП

У зв’язку з критичною екологічною ситуацією, актуальними та перспективними рішеннями є ті, що забезпечують перш за все екологічність та легкість перевикористання.

Створення надійного апаратно-програмного комплексу потребує багато уважності ще на стадії проектування майбутньої системи. Особливу увагу на цьому етапі важливо приділяти обдумуванню технічного обслуговування системи під час її роботи та можливих неполадок у головних елементах системи. Кожен важливий модуль такої системи має бути легко замінений та налаштований у короткий час.

Головна складність у створенні подібних системи – це виявлення помилки, що призвела до краху її роботи, адже помилки можуть бути спричинені як і програмним забезпеченням, так і відмовою фізичних частин комплексу. Тому для систем, головними елементами яких є мікроконтролери, контролери або одноплатні комп’ютери, важливо мати пристрій або установку, яка могла б тестувати їх дієздатність та повідомляти розробника про несправності фізичних частин його системи.

Основною перепоною у створенні універсальної тестувальної станції є великі відмінності у фізичному інтерфейсі, що унеможливлює створення програмно-апаратного комплексу, який би перевіряв функціональність усіх фізичних інтерфейсів пристрою.

Тож було запропоновано дослідити існуючі способи тестування можливостей та характеристик різних контролерів та одноплатних комп’ютерів та на базі досліджень розробити програмно-апаратний комплекс з максимально широким полем застосування та тестування різноманітних моделей.

1 АПАРАТНО-ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ ТЕСТУВАННЯ ТЕХНІЧНОГО СТАНУ АПАРАТНИХ ПЛАТФОРМ

У сучасному світі швидке та якісне тестування є запорукою успішного проекту. У будуванні нових систем, важливо не тільки слідкувати за правильністю імплементованих алгоритмів, але і за станом апаратного забезпечення, функціональністю ключових контролюючих елементів та вчасно виявляти несправності у їх роботі, адже призупинення роботи будь-якої системи несе за собою великі збитки за кожну секунду повної зупинки роботи.

У досить великих і масивних системах із контролерів, які збирають велику кількість інформації та контролюють різноманітні процеси, що забезпечують життєздатність системи, дуже важливо швидко і вчасно визначити чому певний контролер не отримує інформацію або не передає команди до підконтрольних елементів: мати пристрій або станцію, на яку, помістивши контролер, можна перевірити усі виходи та входи контролера та повідомляти клієнта про можливу неполадку у фізичних складових контролера.

Проте головна складність полягає у різноманітності контролерів, кількості їх виходів та особливостях їх розташування у самій схемі. Це значно ускладнює створення універсальної станції, яка могла б працювати з усіма існуючими моделями контролера.

Зокрема відсутність методів визначення несправності апаратної платформи створює екологічну загрозу, оскільки набагато дешевше придбати заміну апаратної платформи, ніж намагатись визначити несправність даного екземпляру. Проте існування тестувальної платформи не тільки дозволяє знайти неполадку розробнику у створеній ним системі, але також дозволяє продовжити життя існуючої плати [1].

Тому було запропоновано дослідити методи тестування апаратних платформ та створити відповідний програмно-апаратний комплекс для їх тестування. Комплекс

має перевіряти функціонування усіх складових контролеру та інформувати користувача про існуючі несправності.

Як результат дослідження, потрібно створити апаратно-програмний комплекс, основними функціями є:

- перевірка функціонування каналів живлення (5V, 3.3V, GND).
- перевірка усіх аналогових та цифрових виходів контролеру на точність передачі та швидкість функціонування.
- перевірка функціонування каналів передачі інформації (UART).
- перевірка внутрішнього пошкодження тепловим скануванням.
- можливість тестування різних апаратних платформ.

1.1 Огляд існуючих систем тестування

На сьогодні виробництво плат та апаратних платформ є одночасно недорогим та швидким, тому контролери виробляють у швидких темпах та величезних кількостях, що значно знижує ціну на данні вироби. Тому тестування плат та контролерів на етапі виробництва проводиться визначення якості виробу, таких як цілісність доріжок на платі, стале закріплення елементів та інших фізичних характеристик.

Такий підхід до виробництва перевіряє фізичну якість зібраної апаратної платформи, проте не перевіряє справжню справність кінцевого виробу, такі як функціонування його входів та виходів, функціонування каналів передачі і т. д.

Тому на даний момент, через складність імплементації не існує реалізацій подібних тестувальних станцій, адже їх розробка є складною і вибагливою для ідеї, яку програми не потребують.

Існування універсальної платформи для тестування плат дозволить вирішити також і екологічну проблему, пов’язану із масовим виробництвом апаратних платформ. Пристрої не використовуються повторно, та в більшості країн не утилізуються правильно, що призводить до бездумного витрачання ресурсів.

Створення станції для тестування плат дозволить визначати несправність та у більшості випадків відновити його працездатність, тим самим скоротити кількість неутилізованої електроніки [2].

1.2 Аналіз існуючих методів тестування апаратних платформ

У процесі пошуку інформації, та аналізу існуючих рішень, було виявлено, що на даний момент не існує реалізацій такого проекту та тестування апаратних платформ зводиться до старих методів тестування.

Одним із методів тестування контролеру є метод мультиметра. Мультиметр – електронний вимірювальний прилад, що поєднує в собі декілька функцій вимірювання: напруги, струму і опору. Існують цифрові і аналогові мультиметри. Тестування проходить наступним чином: щоб перевірити чи сигнал від ніжки контролеру доходить до пункту призначення, потрібно помістити один щуп на ніжку, а інший до точки, яка з’єднується з ніжкою. Якщо мультиметр показує струму між двома точками, можна вважати, що сигнал доходить до свого призначення.

Інший метод тестування та налагодження контролеру є пересилання усіх повідомлень через серійний порт, що дозволить побачити, коли контролер реагує на події в системі, проте такий спосіб не надає достатньо інформації про помилку, зокрема не вказує чи виникає помилка через несправність у системі, чи через несправність устаткування.

1.3 Використання MVC у побудові складних систем

Контролер Model-View (широко відомий як MVC) - це модель дизайну програмного забезпечення, яка зазвичай використовується для розробки інтерфейсу користувача, який розділяє відповідну логіку програми на три взаємопов'язані компоненти. Це робиться для того, щоб відокремити внутрішнє представлення

інформації від того, як інформація подається та приймається користувачем. Цей тип візерунка використовується для розробки макетів сторінок.

Цей метод став популярним для розробки веб-шаблонів, традиційно використовуваних для графічних інтерфейсів користувачів настільних ПК (GUI). Популярні мови програмування, такі як JavaScript, Python, Ruby, PHP, Java, C # і Swift, містять структури MVC, які використовуються для розробки веб-або мобільних додатків.

Даний паттерн складається з наступних елементів:

— Модель: Зразок проміжних компонентів. Це динамічна структура даних програми незалежно від користувальницького інтерфейсу. Він управляє даними безпосередньо, логікою та правилами застосування.
— Вид: Будь-яке представлення інформації, наприклад діаграми, діаграми чи таблиці. Можливі кілька переглядів однієї й тієї самої інформації, наприклад, барні діаграми для управління та представлення таблиць для бухгалтерів.
— Контролер: Приймає введення та перетворює його в команди для моделі чи перегляду.

На додаток до поділу додатків на ці компоненти, конструкція модельного перегляду-управління описує взаємодію між ними.

Модель відповідає за обробку даних у додатку. Він отримує введення користувача від контролера. Вид полягає в тому, щоб представити модель у конкретному форматі. Контролер відповідає на введення користувача та взаємодіє з об'єктом моделі даних. Контролер отримує вхід, додатково перевіряє його, а потім надсилає вхід моделі.

Як і інші розробки програмного забезпечення, MVC виражає фундаментальне рішення проблеми, адаптуючись до кожної системи. Спеціальні конструкції MVC можуть сильно відрізнятися від традиційних описів тут.

Одне з найважливіших розумінь ранньої розробки графічних інтерфейсів користувачів, MVC став одним із перших підходів до опису та впровадження конструкцій програмного забезпечення з точки зору їхніх обов'язків.

Трігве Реенскауг представив MVC у Smalltalk-79 під час відвідування дослідницького центру Xerox Palo Alto (PARC) [11] [12] у 1970-х роках. У 1980-х Джим Алтофф та інші реалізували версію MVC для бібліотеки Smalltalk 80. Лише пізніше стаття 1988 року в журналі The Journal of Object Technology (JOT) висловила MVC як загальну концепцію.

Згодом модель MVC еволюціонувала, викликаючи такі варіанти, як ієрархічний контролер перегляду моделі (HMVC), адаптер перегляду моделі (MVA), менеджер програмного перегляду моделі (MVP), модель-вид-вид (MVVM) та інші, які адаптували MVC до різних контекстів.

Використання шаблону MVC у веб-додатках вибухло популярністю після впровадження WebObjects NeXT у 1996 р., Спочатку написаного в Object-C (який сильно запозичив у Smalltalk) та допомогло застосувати принципи MVC. Пізніше модель MVC стала популярною серед розробників Java, коли WebObjects переносився на Java. Пізніші рамки для Java, такі як Spring (вийшов у жовтні 2002 року), продовжували міцну зв’язок між Java та MVC. Впровадження рамки Джанго (липень 2005 р. Для Python) та Rails (грудень 2005 р. Для Ruby), які зробили сильний акцент на швидкому впровадженні, збільшило популярність MVC поза традиційним корпоративним середовищем, яке давно популярне. Веб-рамки MVC зараз мають великі частки ринку порівняно з веб-інструментами, що не є MVC.

Незважаючи на те, що MVC спочатку був розроблений для настільних обчислень, він широко використовувався як дизайн для світових веб-додатків у великих мовах програмування. Для реалізації схеми було створено кілька рамок сітки. Ці програмні рамки в основному мають сенс у розподілі обов'язків MVC між клієнтами та серверами.

Деякі мережі MVC-фреймворків застосовують тонкий клієнтський підхід, який розміщує майже всю логіку моделі, перегляду та управління на сервері. Це відображено в таких структурах, як Zango, Rails та ASP.net MVC. При такому підході

клієнт або надсилає запити на гіперпосилання, або подає формуляр для подання модератору, а потім отримує повну та оновлену веб-сторінку (або інший документ) з подання; Модель повністю існує на сервері. Інші рамки, такі як Angular JS, AmberJS, JavaScript MVC та Backbone MVC, дозволяють компонентам частково виконувати функції клієнта (див. Також Ajax).

MVC відключає різні компоненти програми, тому розробники можуть працювати паралельно з різними компонентами, не впливаючи і не блокуючи один одного. Наприклад, команда може розділити своїх розробників між передніми і зворотними. Резервні розробники можуть структурувати дані та спосіб взаємодії користувачів з ними, не доповнюючи інтерфейс користувача. На відміну від цього, розробники, що розробляються, здатні розробляти та тестувати додатки до того, як структура даних стане доступною.

Один і той же (або подібний) вигляд для однієї програми може бути відображений для іншої програми з різними програмами, оскільки цей погляд обробляє спосіб відображення даних користувачеві. На жаль, це не працює, коли код корисний і для введення коду. Наприклад, DOM-код (зі спеціальними рефератами програми) корисний як для графічного відображення, так і для введення користувача. (Зауважте, що незважаючи на назву моделі об'єктного документа, DOM насправді не є моделлю MVC, оскільки для користувача є інтерфейс програми).

Для вирішення цих проблем MVC (і подібні зразки) зазвичай поєднуються з архітектурою компонентів, яка забезпечує набір компонентів інтерфейсу користувача. Кожен компонент інтерфейсу - це компонент високого рівня, який поєднує три необхідні компоненти MVC в єдиний пакет. Створюючи незалежні один від одного компоненти високого рівня, розробники можуть швидко та легко використовувати компоненти в інших додатках.

2 АРХІТЕКТУРА АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

Аналізуючи поставлену задачу та методи її реалізації, було вирішено створити апаратно-програмний комплекс, який умовно можна розділити на апаратну та програмну частини.

Апаратну частину було вирушено розробляти на основі платформи Orange Pi та Arduino Mega, застосовуючи мову C++. Веб-додаток вирішено розробляти за допомогою веб інструментів мови Java.

2.1 Вибір архітектури апаратної частини

Після аналізу поставленої задачі було розроблено наступне архітектурне рішення: апаратно-програмний комплекс складається з головної плати, яка вміщує сервер та головний контролер, та модулю розширення для тестування платформ. Зв’язки між елементами цієї архітектури зображені на рисунку 3.1.

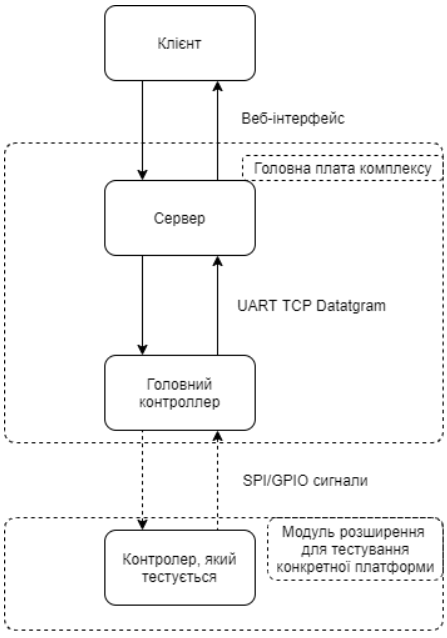

Рисунок 3.1 – Архітектура апаратно-програмного комплексу

Головним центром апаратно-програмного комплексу є сервер. У ньому зосереджена основна бізнес-логіка. Сервер обробляє дані від клієнта, та формулює команди для головного контролера. Також він обробляє отримані результати дослідження від нього, та передає сформовану статистику для відображення клієнту.

Головний контролер під'єднаний до каскаду мультиплексорів, через які контролює підключений контролер для тестування. Мультиплексори – пристрої передачі цифрової інформації, які здійснюють комутацію одного з декількох інформаційних входів до одного виходу. Мультиплексори мають декілька інформаційних входів, адресні входи, вхід дозволу мультиплексування (стробуючий вхід) та один вихід. Кожному з інформаційних входів мультиплексора відповідає номер, який називається адресою, двійкове число якого подається до адресних входів. За допомогою каскадів цих мультиплексорів можна доступитись до ніжок контролерів різних конфігурацій та видів, що забезпечує приладу універсальність у використанні.

Під час користування програмою, користувач взаємодіє з клієнтським додатком, яким є веб-сайт в даному випадку. На рівні користувача реалізований інтерфейс, за допомогою якого відбувається налаштування програми та перегляд результатів роботи. Також на користувацькому рівні відбувається попередня обробка даних перед відправленням на сервер і також відображення роботи сервера.

2.2 Опис архітектури серверу

Для відповідності сучасним течіям розробки програмного забезпечення, серверний застосунок було розроблено використовуючи шаблон проектування MVC.

Model-View-Controller (MVC, рисунок 3.2.1) - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, представлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

- Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан .
- Представлення (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.
- Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

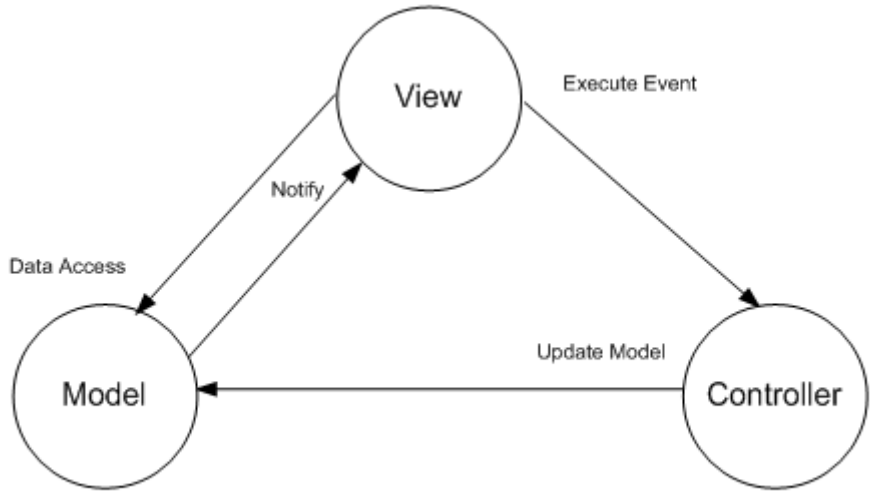

Figure 4: Simplistic MVC

Рисунок 3.2.1. Спрощена схема роботи шаблону MVC

Використання цього шаблону проектування дозволяє швидко розробляти програмне забезпечення, залишаючи кінцевий продукт простим до змін та покращення.

2.3 Опис архітектури веб-додатку

Веб-додаток використовує сучасну технологію PWA для забезпечення користувача як можливістю користуватись системою через веб-браузер, так і використовуючи застосунок. Саме фреймворк Vaadin забезпечує просту реалізацію подібних застосунків.

Прогресивний веб-додаток, або PWA - технологія в web-розробці, яка візуально і функціонально трансформує сайт в додаток (мобільний додаток в браузері).

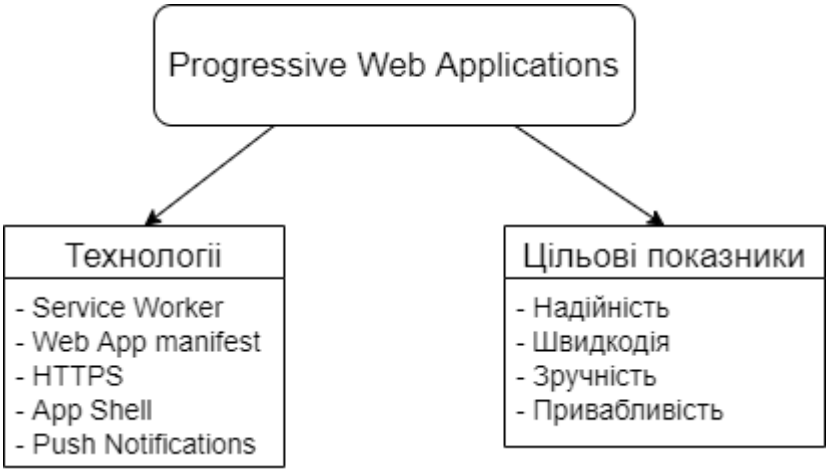

Рисунок 3.3.1. Схема особливостей PWA

PWA є гібридним рішенням і дозволяє відкрити програму за допомогою мобільного браузера. При цьому повністю зберігається функціонал застосунку (Рисунок 3.3.1):

- відправка push-повідомлень;
- робота в режимі офлайн;
- доступ до апаратного забезпечення пристрою (з обмеженнями);
- установка ярлику (іконки) на робочому столі мобільного пристрою, що візуально не відрізняється від ярлика нативного додатки, тощо.

Такий застосунок спілкується з сервером використовуючи REST API.

3 АНАЛІЗ ТА ОБҐРУНТУВАННЯ РЕАЛІЗАЦІЇ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

Під час розробки програмного продукту була використана модель бізнем логіки. Такий підхід дозволяє підвищити ефективність роботи складної системи протягом наступного життєвого циклу програмного продукту без витрат пов’язаних з перепроектуванням. Також такий підхід дозволяє паралельно працювати багатьом програмістам.

Після вирішення проблеми були створені основні критерії, яким відповідає програмна частина системи:

— Архітектура апаратно-програмної платформи має забезпечувати можливість швидкого нарощування функціоналу
— Інтерфейс користувача та взаємодія з ним мають бути стійкими до розширення можливостей системи

Найефективнішим методом побудови програмного забезпечення з подібними вимогами є об’єктно-орієнтоване програмування.

Об'єктно-орієнтоване програмування є спадкоємцем процесуального (структурного) програмування. Процедурне програмування описує програми як групи багаторазових кодових одиниць (процедур), які визначають вхідні та вихідні параметри. Процедурні програми складаються з процедур, які викликають одна одну.

Проблема процедурного програмування полягає в тому, що використання коду є важким і обмеженим - лише процедури можна повторно використовувати, і важко зробити їх загальними та гнучкими. Немає простого способу роботи з абстрактними структурами даних з різними реалізаціями.

Об'єктно-орієнтований підхід спирається на парадигму, що кожна програма працює з даними, що описують сутності (об'єкти чи події) з реального життя. Наприклад: системи програмного забезпечення бухгалтерського обліку працюють з

рахунками-фактурами, предметами, складами, доступністю, замовленнями на продаж тощо.

Так з’явилися об’єкти. Вони описують характеристики (властивості) та поведінку (методи) таких утворень реального життя.

Основні переваги та цілі ООР - зробити складне програмне забезпечення швидшим для розвитку та легшим у обслуговуванні. ООР дозволяє легко використовувати код, застосовуючи прості і широко прийняті правила (принципи). Давайте перевіримо їх.

Для того, щоб мова програмування була орієнтована на об'єкти, вона повинна забезпечувати роботу з класами та об'єктами, а також реалізацію та використання основних об'єктно-орієнтованих принципів та концепцій: успадкування, абстрагування, інкапсуляція та поліморфізм. Давайте підведемо підсумки кожного з цих основних принципів ООП:

— Інкапсуляція
— Наслідування
— Абстракція
— Поліморфізм

Деякі теоретики ООП також розглядають концепцію поводження з винятками як додатковий п’ятий фундаментальний принцип ООП. Ми не будемо вступати в детальну суперечку щодо того, чи є виключення частиною ООП, а скоріше зазначимо, що винятки підтримуються у всіх сучасних об'єктно-орієнтованих мовах і є основним механізмом обробки помилок та незвичних ситуацій в об'єктно-орієнтованому програмуванні.

3.1 Оцінка вимог

Після аналізу архітектури системи та її головних функцій, було визначено головні вимоги до створюванної станції тестування апаратних пристроїв:

— визначати модель плати кількістю та розташуванням ніжок плати
— достукатись до кожної ніжки, відправляти та отримувати сигнали ніжки
— демонструвати поточний робочий стан апаратного пристрою
— формувати звіт про виконану перевірку, який матиме детальний опис роботоздатності кожної ніжки

Для демонстрації роботи системи було обрано дві плати: Arduino Nano та Arduino Mega. Обидві плати були попередньо перевірені ручним шляхом, непрацюючі ніжки були позначені та занотовані.

3.2 Вибір мікроконтролера

Arduino Mega 2560 - це мікроконтролер ATmega2560 (інформаційний лист). Він включає в себе все, що вимагає роботи, конкретний мікроконтролер 54 виходи цифровий / виробничий (15 можна використовувати як вихід ШІМ), 16 можливих аналогових, 4 UART (доступ до загального обладнання), перемикач квотатора на 16 МГц, перетворювач USB, джерело живлення, програмний перемикач програмного забезпечення ICSP та кнопка скидання. Щоб розпочати роботу на своєму пристрої, просто підключіть адаптер змінного струму / постійного струму або акумулятор або підключіть його до комп'ютера за допомогою кабелю USB. Mega Arduino сумісний з більшістю карт розширення, розроблених для Arduino Duemilanove і Diecimila.

Mega 2560 - це оновлена версія Arduino Mega. Arduino Mega 2560 відрізняється від усіх попередніх плат тим, що використовує мікроконтролери ATmega16U2 (ATmega8U2 типу R1 та R2) для перетворення інтерфейсу USB-UART у мікросхему FTDI.

Рис. 3.1 Arduino Mega 2560

На платі Mega 2560 версії R2 доданий резистор, що підтягує до землі лінію HWB мікроконтролера 8U2. Такий розмір дозволяє спростити процес оновлення прошивки та перетворити пристрій у режим DFU.

Зміни на платі R3 наведені нижче:

— Pinout 1.0: Додано шпильки SDA та SCL (близькі до штифта AREF), а також два нові штифти, розташовані біля стандарту RESET. Перший - IOREF - дозволяє розширювальним платам адаптуватися до робочої напруги Arduino Цей модуль передбачений для сумісності з картами розширення як для 5В Arduino-базованих мікроконтролерів AVR, так і 3,3 В Arduino Причини. Другий продукт не пов'язаний і зберігається для майбутніх цілей.

— Підвищення захисту голосу в циклі переселення.

— Мікроконтролер ATmega16U2 замінено на 8U2.

Arduino Mega 2560 має знімні запобіжники, що захищають порти USB від коротких і надмірних портів. Хоча більшість комп'ютерів мають власний захист, такі запобіжники забезпечують додатковий захист. Якщо на USB-порту використовується більше 500 мА, запобіжник автоматично включається до короткого замикання або надмірного навантаження.

Максимальна довжина та ширина Mega2560 - відповідно 10,2 см та 5,4 см, враховуючи USB-з'єднання та джерело живлення від карти. Три кріпильні отвори дозволяють розмістити пил на поверхні або рамі. Зауважте, що відстань між цифровими штифтами 7 та 8 не настільки велика, як звичайні 2,54 мм та 4 мм.

Arduino Mega2560 розроблений так, щоб відповідати більшості карт розширення від Arduino Uno, Diecimila та Duemilanove. Для цього цифрові штифти мають 0-13 (а також штифти AREF та GND поруч із ними), аналоговий роз'єм 0-5, електричний роз'єм та роз'єм ICSP на тих же платах. На додаток до вищевказаного блоку, основні лінії приймачів UART з'єднані з тим же терміналом (0 і 1), що і зовнішні лінії 0 і 1 (клеми 2 і 3 відповідно). Лінії SPI підключені до з'єднання ICSP двох панелей - Mega2560 і Duemilanove / Diecimila. Слід пам’ятати, що в Argaino Mega положення штифтів I2C відрізняється між платами Duemilanove / Diecimila: Arduino Mega, це 20 і 21 штифт, і Duemilanove / Diecimila, аналоговий 4 і 5 вхід.

Arduino Mega 2560 пропонує кілька варіантів зв’язку з комп'ютером, іншим Arduino або іншими мікрофонами. ATmega2560 має чотири передавачі UART, призначені для роботи в круговому напрямку (зі стандартним TTL 5V). Мікроконтролер ATmega16U2 (або ATmega8U2 на кабелях R1 і R2) забезпечує з'єднання між передавачем і USB-комп'ютером USB, а при підключенні до ПК Arduino можна визначити як порт комп'ютера, який називається (для цього операційна система Windows вимагає, щоб INF-файл повинен відрізнятися від OSX та Linux, де ідентифікатор панелі як COM-порт є автоматичним). Комп'ютерний пакет Arduino містить спеціальну програму SerialMonitor, яка дозволяє читати та надсилати текстові дані до Arduino. Під час передачі даних на мікросхему ATmega8U2 / ATmega16U2 під час використання USB-кабелю на платі будуть блимати індикатори

RX та TX. (При передачі послідовних даних на штирі 0 і 1 без використання USB-перетворювача ці дроти не працюватимуть).

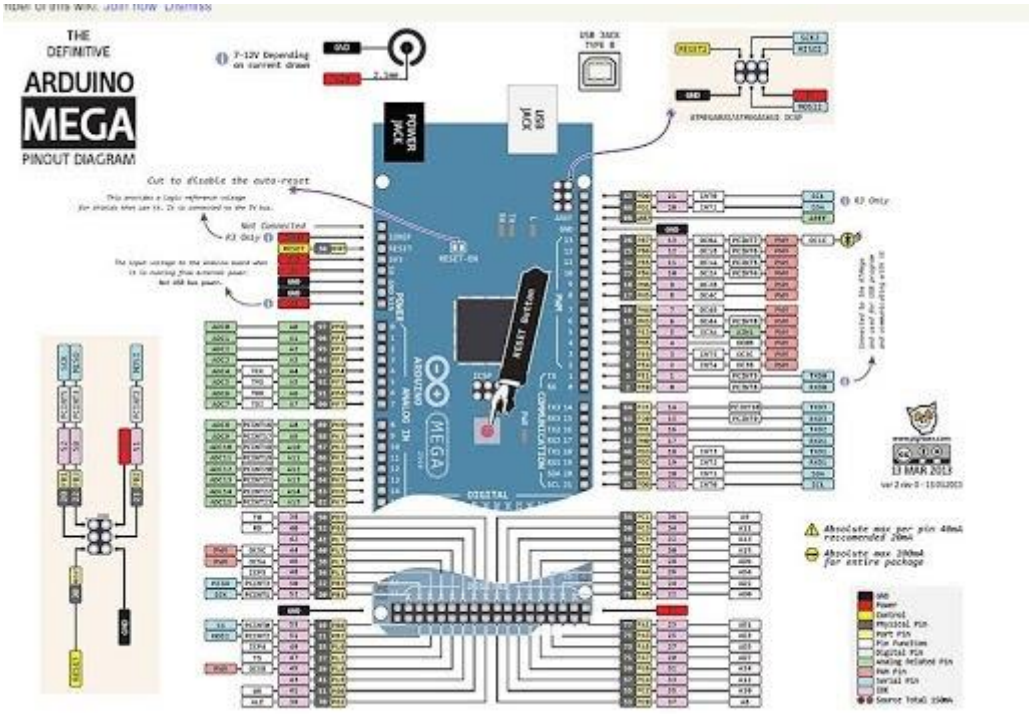

Рис. 3.2 Схема розташування пінів Arduino Mega 2560

Бібліотека SoftwareSerial дозволяє послідовно спілкуватися для кожного продукту Mega2560.

Мікроконтролер ATmega2560 управління також забезпечує апаратну підтримку зв'язку TWI та SPI. Програмне забезпечення Arduino включає бібліотеку Wire, що полегшує роботу на шині TWI; Додаткову інформацію див. У бібліотеці потоків. Використовуйте бібліотеку SPI для роботи з програмою інтерфейсу SPI.

3.3 Вибір мультиплексора

Мультиплексор - це блок з декількома вхідними елементами, одним або декількома елементами управління, і одним виходом Контролер дозволяє передавати сигнал вводу і виходу; вибір потрібної ідеї здійснюється за допомогою правильної комбінації керуючих сигналів.

Аналогові та цифрові багатоаналітичні файли істотно відрізняються від принципу роботи. Перший в електронному вигляді підключає вибраний розподіл до виходу (з мінімальним опором між - близько сотень чи десятків Ом). Останній не здійснює прямого електричного з'єднання між входом і виходом, а лише "копіює" вихідний рівень ("0" або "1") вибраного входу. Аналогові протоколи іноді називають обертанням або обертанням.

Одиниця проти множення називається демультиплексором. У разі аналогових аналогових файлів (з використанням передавачів на місцях) різниці між множинними та демультиплексорами немає; Такі пристрої можна назвати альтернативними.

Мультиплексори можна використовувати для розподілу частоти, блоків стимуляції, блоків передачі та інших. Мультиплексори можна використовувати для перетворення паралелі в десяткову. Для такої змінної достатньо застосувати паралель до агрегування інформації в декількох файлах і відзначати адреси, що використовуються в послідовності, що вихід підключається до черги виводу, від першого до кінця. останній.

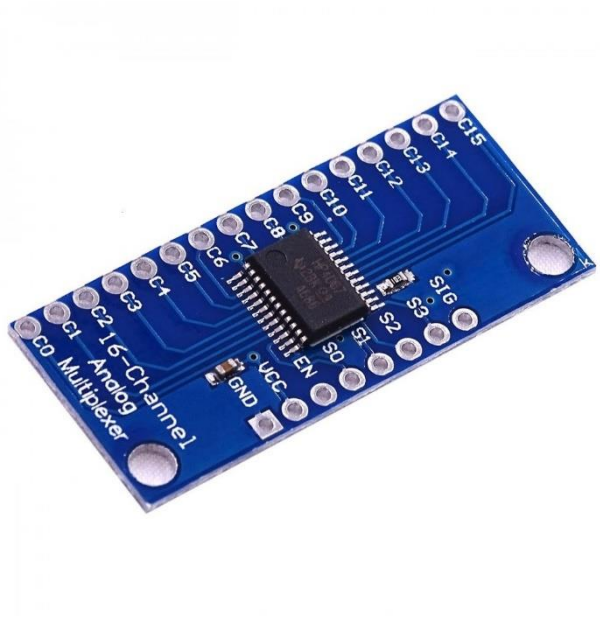

Рис. 3.3 Мультиплексор hc4067

74HC4067; 74HCT4067 - це однополюсний аналоговий вимикач на 16 хв (SP16T), який підходить для використання в аналогових або цифрових програмах

мультиплексора / демультиплексора 16: 1. Комутатор має чотири цифрові входи (S0, S1, S2 і S3), шістнадцять незалежних входів / виходів (Yn), загальний вхід / вихід (Z) і цифровий вхід включення (E). Коли E високий, вимикачі вимикаються. Вхідні дані включають затискаючі діоди. Це дозволяє використовувати резистори обмеження струму для інтерфейсних входів до напруг, що перевищують VCC.

3.4 Програмування мікроконтролеру

Arduino Mega була запрограмована за допомогою програмного забезпечення Arduino (завантажити). Для отримання додаткової інформації дивіться довідку та приклади.

ATmega2560 Arduino Mega оснащений утилітою завантажувача, яка дозволяє завантажувати нові програми на мікрофон без необхідності використання зовнішнього додатка. Інтеграція виконується за стандартним протоколом STK500 (опис, адреса C-файлу).

Однак мікрофон можна також пом’якшити за допомогою з'єднання ICSP (In-Circuit Serial Programming), не ігноруючи балансу навантаження; Для відповідних інструкцій див. Відповідні інструкції.

Вихідний код мікропрограмного забезпечення мікроконтролера ATmega16U2 (або ATmega8U2 на треках R1 і R2) знаходиться в реле Arduino. Прошивка мікропрограмного забезпечення ATmega16U2 / 8U2 включає в себе програмне забезпечення оновлення мікропрограмного забезпечення (DFU), яке дозволяє оновлювати вбудовані програми. Щоб увімкнути режим DFU, потрібно:

— Для версій R1: Закрийте бункер на задній панелі плати (біля італійського вигляду), а потім встановіть на 8U2.
— Для версій R2 і вище, щоб полегшити перехід до режиму DFU, існує стійкість потягування лінії HWB на наземних мікрофонах 8U2 / 16U2. Після переходу в режим DFU ви можете використовувати програмне

забезпечення FLIP (Windows) або DFU Atmel (для завантаження нового обладнання) для Mac OS X та Linux). Необов'язковий варіант - увімкнути мікрофон як внутрішній програміст провайдера та використовувати зовнішню програму у випадку, якщо джерело живлення DFU вимкнено. Додаткову інформацію див. У посібнику користувача.

Тому щоразу, коли ви завантажуєте програмне забезпечення, не потрібно натискати кнопку скидання, Arduino Mega 2560 розроблений як спосіб завантажити його з підключеного комп'ютера. Один із штифтів ATmega8U2, що бере участь у керуванні потоком даних (DTR), підключений до контакту RESET мікроконтролера ATmega2560 через конденсатор 100 nF. Коли на рядку DTR з'являється нуль, штир RESET також має достатньо низький час, щоб перезапустити мікрофон. Ця функція використовується для включення мікроконтролера одним клацанням на кнопці сайту програми Arduino. Така конструкція дозволяє скоротити час завантажувача, оскільки версія прошивки завжди працює на сигналі DTR.

Однак ця процедура може призвести до інших наслідків. Коли ви підключите Mega 2560 до комп'ютера, на якому працює Mac OS X або Linux, мікрофон буде скидатися кожен раз, коли програмне забезпечення підключено до плати. Після скидання Arduino Mega2560 завантажувач працює близько півсекунди. Хоча завантажувач призначений для ігнорування сторонніх даних (наприклад, усіх даних, не пов'язаних з версією мікропрограмного забезпечення нової програми), він може заважати першим кільком цифрам з пакету, що надсилається на плату, як тільки з'єднання встановлено. Отже, якщо запущене програмне забезпечення Arduino забезпечує прийом налаштування або інші дані з комп'ютера на початку, переконайтеся, що програмне забезпечення, яке взаємодіє з Arduino, надсилається до однієї секунди після встановлення з'єднання.

Існує пакет Mega 2560 (з написом RESET-EN), який, розблокувавши його, можна вимкнути автоматичним скиданням мікроконтролера. Для відновлення процесу переміщення автомобіля необхідно продати обробку, що лежить на

порожнистих краях. Автоматичне скидання також можна вимкнути, підключивши резистор 110 Ом між штифтом RESET і 5V.

3.5 Одноплатний комп’ютер Orange Pi

Orange Pi - недорогий конкурент Raspberry Pi. Він розроблений китайською компанією Software Software Shenzhen Xunlong CO, що базується в місті Шеньчжень у провінції Гуандун. Як і Raspberry Pi, Orange Pi - це проект з відкритим кодом. На відміну від Raspberry Foundation, який вирішив запропонувати дві моделі (якщо не рахувати щорічних змін), Orange Pi пропонує широкий асортимент карт. Це велика перевага, оскільки ви можете вибрати карту, яка найкраще підходить для її застосування. Однак це не завжди просто, якщо знайти їх у конфесіях. Ось кілька діаграм порівняння, які допоможуть вам знайти карту, яка відповідає вашим потребам з Raspberry Pi 3.

Переважна більшість IP-карт Orange працює з мікропроцесором H3 (Cortex - A7) з тактовою частотою 1,2 або 1,6 ГГц. У записі Pi Zero використовується H2 тактова частота 1, 2 ГГц. Це дійсно невелика картка для всіх (Ethernet, WiFi, GPIO), ідеальна для невеликих об'єктів. Це також єдина модель в асортименті, що пропонує живильний кабель Ethernet (POE). Пошкодуйте тільки Orange Pi Zero за відсутність роз'єму камери (формат CSI), тому вона повернеться до камер (або веб-камери) USB.

Ще одна сильна сторона моделей Orange Pi, декілька карт систем 8-16 ГБ флеш-пам’яті, що дозволяє встановлювати систему безпосередньо на карту або використовувати додатковий простір для зберігання.

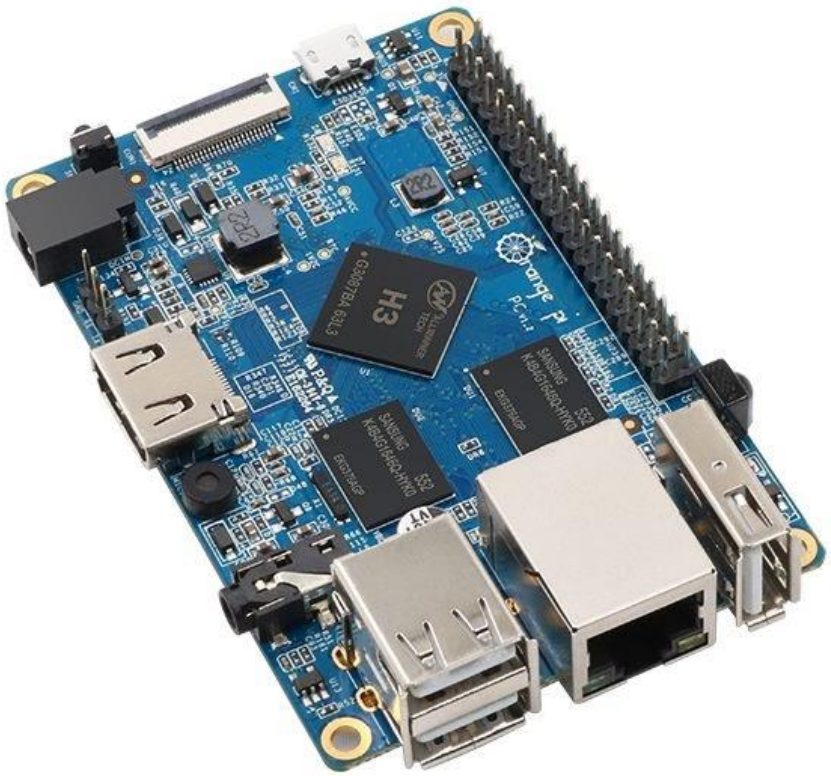

Рис. 3.3 Одноплатний комп’ютер Orange Pi

Переваги плати:

— Надзвичайно дешево за те, що обіцяє: Orange Pi PC надзвичайно дешевий, особливо в порівнянні з SBC, за які він конкурує.
— Велике співвідношення ціни та якості
— Підтримує майже всі ОС, які підтримуються Raspberry: Orange Pi підтримує Raspbian, Ubuntu, Android та багато інших операційних систем. Він стверджує, що підтримує всі ОС, які підтримуються Raspberry Pi, і, здається, це правда більшість часу.

Недоліки плати:

— Підтримка ОС першої сторони практично не існує

— Має деякі проблеми з термічним дроселюванням: Orange Pi може сильно нагрітися, аж до тих пір, коли йому потрібно відключити серцевини, щоб не замикатися.
— Має проблеми з повільними картами microSD: Що-небудь менше, ніж карта TF/microSD класу 10, ймовірно, не працюватиме з Orange PI. Оскільки тестування та виробництво Orange Pi масово поспішало, то в результаті він не зміг бути більш сумісним із більш повільними картами SD. Хоча слід зазначити, що це не є великою проблемою, оскільки ви можете отримати карти microSD класу 10 досить дешево в наш час.
— Немає відеороз'єму: Потрібно використовувати RPIO, еквівалентний для передачі будь-якого відео.

3.6 Програмування одноплатного комп’ютеру

Для легкої розробки та возз’єднання плати Orange Pi необхідно створити операційну систему. Orange Pi підтримує понад 15 операційних систем. Armbian був обраний для розробки.

Armbian - це основна операційна система для планшетних комп'ютерів, на яку можуть покладатися інші проекти. Це тонкий дистрибутив Debian або Ubuntu, який спеціалізується на розробці плат ARM. Кожна система інтегрує, координує та вдосконалює Armbian Architecture. Він має потужні інструменти для створення та розробки програмного забезпечення для створення вашої колекції.

Основні переваги використання проекту:

— Офіційна підтримка плати Orange Pi, оскільки вона гарантує надійну роботу системи.
— На основі UBUNTU16
— Є багато програм, які розробляють програми
— Ефективна компоновка обладнання Orange Pi

— Ethernet-адаптер з DHCP-сервером і SSH готовий до прямого підключення (22)
— Бездротовий адаптер DHCP доступний зараз, але він вимкнено. Ви можете використовувати конфігурацію рук, щоб підключитися до маршрутизатора або зробити AP
— Включає NAND, SATA, eMMC та USB-текст (nand-sata-install)
— Оновлення здійснюються за допомогою відповідного методу оновлення рівня
— У тексті журналу активності відображаються: назва примітки, поштова станція, оригінальна версія, система запуску, години, пам’ять, IP-адреса, швидкість процесора, рівень диска, локальна температура, якщо вихід, використання SD-карти, коефіцієнт заряду акумулятора та кількість оновлень для встановлення

4 ЗАСОБИ РОЗРОБКИ

Для розробки даного програмно-апаратного комплексу використовувалось рызноманітні технології розробки за допомогою мікроконтролерів, одноплатних комп’ютерів та інші. Було проаналізовано вимоги кожного етапу розробки та вибрано наступні засоби розробки програмного забезпечення.

4.1 Jetbrains IDEA

Intelligence IDEA - це інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, розроблене конкретними Java, JavaScript, Python, Jet Brains.

Перша версія з'явилася в січні 2001 року і набула популярності в середовищі Java за допомогою широкого спектру інтегрованих інструментів рефакторингу [8], що дозволило програмістам швидко переробити вихідний код програми. Дизайн середовища орієнтований на продуктивність програміста, тому ви можете зосередитися на функціональних завданнях, тоді як Intelligence IDEA піклується про звичайні завдання.

Початок роботи з шостою версією продукту Intel J IDEA надає інтегровані інструменти для розробки графічних інтерфейсів користувачів. Серед інших особливостей, середовище сумісне з багатьма популярними безкоштовними інструментами для розробників, такими як CVS, Subversion, Apache Ant, Maven та Junit. У лютому 2007 року розробники Intelligence випустили початкову версію плагіна для підтримки програмування Ruby.

З версії 9.0 середовище доступне у двох версіях: спільноті та кінцевій версії. Community Edition - це абсолютно безкоштовна версія, доступна за ліцензією Apache 2.0, вона підтримує повну інтеграцію з Java SE, Kotlin, Grovy, Scala, а також найпопулярнішою системою управління версіями. Доступний у комерційних

ліцензіях, Ultimate Edition включає підтримку Java EE, графіки UML, розрахунки покриття коду, а також інші системи контролю версій, мови та рамки.

Дане середовище розробки програмного забезпечення має такі переваги:

- Інтелектуальне автоматичне доповнення, інструменти аналізу якості коду, проста навігація, вдосконалене рефакторинг та форматування для Java, Groovy, Scala, Closeor та Erlang.
- Професійний інструментарій для розробки додатків для Android.
- Підтримка JavaFX 2.0, інтеграція з SeanBuilder; Дизайнер інтерфейсу для гойдалок.
- Інтеграція з автоматизованими інструментами управління будівництвом та управлінням проектами, включаючи Maven, Gradle, Ant та інші.
- Інструменти тестування з підтримкою Junit, TestNG, Spock, Skelatest та Spec2.
- Інтеграція з системами управління версіями, включаючи Git, Subversion, Mercury та CSV.

4.2 Arduino IDE

Інтегроване середовище розробки Arduino (IDE) - це кросплатформенне додаток (для Windows, macOS, Linux), написане на функціях C та C ++ . Він використовується для запису та запису програм із сумісними з Arduino дошками, але також, за допомогою сторонніх мостів, інших комітетів з розвитку.

Вихідний код для IDE був випущений на публічній ліцензії GNU типу 2. Arduino IDE підтримує мови C та C ++ за допомогою спеціальних правил кодування. Ідентифікатор Ardeino IDE забезпечує бібліотеку даних від проекту Wire, що пропонує широкий спектр практичних та виробничих методів. Код, призначений

лише для користувача, потребує двох основних функцій для запуску зображення та циклу основної програми, які складаються спільно з ланцюжком інструментів GNU, а також іншої, що входить до дистрибутиву IDE. Arduino IDE використовує програмне забезпечення avrdude для перетворення коду дії, який може бути перетворений у текстовий файл у шістнадцятковий код, вставлений у плату Arduino в програмі завантаження компанії. При дотику avrdude використовується як інструмент фільтра для генерації коду користувача для планшетів на дошках Arduino.

Зі зростанням популярності Arduino як комп’ютерного пристрою інші виробники починають впроваджувати компактні комп'ютери з відкритим кодом та сердечники, які можуть створювати та встановлювати інші MCU, які не підтримують прямий доступ. У жовтні 2019 року асоціація Arduino незабаром випустила нову IDE: Arduino Pro та іншими розширеними функціями.

4.3 Eclipse для Arduino

Arduino C++ IDE доступний на ринку Eclipse. Для початку користувачі повинні завантажити Eclipse для C ++ зі сторінки завантажень Eclipse. C ++ IDE можна встановити за допомогою нового інсталятора Eclipse, або пакет C ++ можна завантажити безпосередньо напряму. Після налаштування клієнта Marketplace, його можна використовувати для пошуку та встановлення Arduino IDE C ++. На ринку є інші Arduino-середовища, які позначені логотипом CDT.

Arduino C++ IDE є досить повною IDE. Його можна для створення демонстраційних матеріалів, а також використовувати потужність IDE Eclipse C / C ++ для проектів різних рівнів складності.

Найбільш популярна функція - це підтримка розвитку бібліотек Ардуїно. Вона містить величезну колекцію, яка постійно розширюється. Нажаль існуючі IDE не всі підтримують роботу з данною функцією. Оскільки дані бібліотеки є напрацюванням різних користувачів Arduino, такий код є ненадійним, адже компанія не верифікує та не тестує цей код власноруч, тому при його використанні потрібно

бути досить обережним. Проте з іншого боку, такий спосіб роботи як «агрегатора» коду стимулює розвиток ком’юніті, надаючи усім користувачам можливість реалізовувати проекти простіше та швидше.

4.4 UART

У перекладі з англійської мови UART – "Універсальний асинхронний приймач".

Кожен біт кожного байту передається за той самий період розподілу (фактично часовий проміжок). Кількість даних у пакеті становить 8 байт, але крім даних, кожен пакет також містить службову інформацію:

- Початковий біт (обов’язково)
- Стоп біти (також обов'язковий, можна використовувати 1, 1.5, 2 стоп-біта)
- Біти парності (необов’язково, існують парні та непарні)

Так як інтерфейс асинхронний, то більшої значущості має швидкість передачі даних - і у приймача, і у передавача вона повинна бути однаковою.

Швидкість вимірюється в бітах в секунду, або коротко - в бодах. Стандарт RS232 має на увазі швидкості від 1200 до 115200 бод, хоча по факту існують швидкості і нижче, і вище, причому до десятків мегабод!

Зрозуміло, точність всюди відносна і швидкість ніколи не буде дорівнювати 9600 бодам з точністю до одиниць. Стандарт передбачає можливу помилку в швидкості до 5% (не більше 3% для впевненого прийому).

Всі плати Arduino, побудовані на основі оригінальних, мають мінімум один інтерфейс UART, просунуті же плати, типу Arduino Mega 2560 Або Arduino Due, маю одразу 4 апаратних інтерфейсу! Вони не завантажують контролер, так як вони

відокремлені від ядра; все, що необхідно - це конфігурувати порт і запхати дані в буфер, після чого операції передачі підуть незалежно від вас.

Звичайно, існують і програмні реалізації UART, але вони навантажують процесор. У будь-якому випадку, краще використовувати спочатку апаратні інтерфейси, а потім вже починати придумувати щось програмне.

Контролери Arduino використовують логічні рівні такі ж, яким є харчування, тобто для найпопулярнішою плати Arduino UNO рівні дорівнюватимуть - нуль = 0В, 1 = 5В.

Висновки підключені до перетворювача інтерфейсів через резистори з опором 1 КОм, а до гребінок з боків плати - безпосередньо, тому сигнали з гребінок матимуть більший пріоритет. Періодично це заважає прошивати плати з підключеним датчиком по UART, так як для прошивки теж використовується UART.

Мікросхема перетворювача інтерфейсів не робить з себе ще один COM-інтерфейс для комп'ютера, вона лише емулює його. Незважаючи на це, всі програми, які працюють з COM-портом за допомогою Windows API (Win32 API), які не відрізнять порт від фізичного порту комп'ютера.

За час існування цифрових технологій було створено десятки послідовних протоколів. USB (універсальна досліджувана шина) та Ethernet - це пример двох найбільш популярних зараз послідовних протоколів. Інші дуже популярні послідовні інтерфейси - це SPI, I2C та послідовний інтерфейс, який реалізує в цій стадії. Каждый їх інтерфейс можна віднести до однієї з двох подгрупп - Асинхронные і Синхронные.

У цілому протоколи передачі даних поділяються на дві великі групи - паралельні та послідовні.

Паралельні інтерфейси передають одночасно (паралельно) декілька біт інформації (відсутня, власно і їх назва). Для передачі даних таких інтерфейсів потрібно мати доступні шини, що знаходяться з 8, 16 або більше провідників.

Послідовні інтерфейси передають по одному біту за раз. Теоретично такий інтерфейс може працювати на однорідному єдиному проводі. На практиці використовується до чотирьох.

Синхронний протокол завжди включає лінію тактового сигналу. Це забезпечує більш високий простір (і зачастує більшу структуру) передачі даних, але не вимагає, як мінімум один додатковий провід. Принтер синхронних інтерфейсів - це SPI та I2C.Асинхронний інтерфейс підключає, що передаються без підтримки підключеного тактового сигналу. Цей метод передає ідеально підходить для мімізації кількості проводуючих, але це важливо, но для надійної передачі та даних даних, що потрібні додаткові додатки. Досліджений інтерфейс, який ми обсудимо в цій стадії, є найбільш розповсюдженим і старим асинхронним і протокольним. Часто виконує так, що, коли людина говорить «послідовний», він має в своєму роді саме цей протокол.

Асинхронний послідовний інтерфейс, з яким тут ідеальна речь, широко використовується у вбудованих системах. Якщо ви хочете, додайте у свій проект модуль GPS, Bluetooth, XBee, послідовні ЖК-дисплеї або багато інших інших вбудованих пристроїв, вам, мабуть, пропонується надіслати стіл з одним з постійних інтерфейсів.

Асинхронний послідовний протокол має ряд вбудованих правил - механізмів, які допомагають забезпечити надійну та безошибочну передачу даних. Це механізми, які дозволяють передати дані без використання зовнішнього тактового сигналу:

— Біти даних
— Біти синхронізації
— Біти перевірки четности
— Скорость передачі

Завдяки сумісності цих правил - віртуально, протокол дуже гнучкий. Для успішної зв’язки необхідно перемогти, що налаштовує на широкі налаштування на використанні одинакових правил.

Універсальний асинхронний прийом / перехідчик (UART) представляє собою єдиний блок схеми, відповідальний за реалізацію послідовної зв'язку. По суті, UART показує в якості посередника між паралельними та послідовними інтерфейсами. У

одної частини UART є шина з восьми (або навколо того часу) лінійних даних (плюс деякі керовані контакти), з другою - два наступних пристрою - RX і TX.

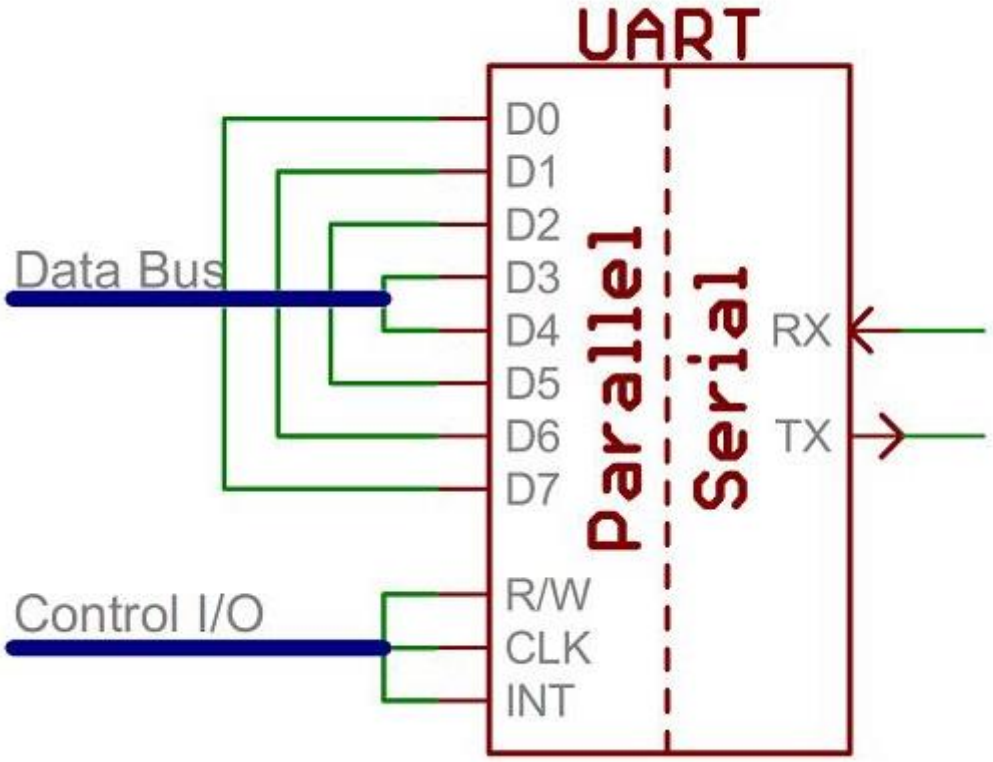

Рис. 4.1 Спрощена схема реалізації UART передачі

Інтерфейси UART фактично бачать відокремлені мікросхеми, но чаще всього вони вбудовані в мікроконтролери. Якщо ви дізнаєтесь, то у вашому МК протокол UART, ви можете прочитати дані про цей контролер. У деяких немає ні одного, у деяких є, у деяких їх кілька. Наприклад, Arduino Uno, заснований на старій добрій ATmega328, має лише один UART, в той час як Arduino Mega - побудований на ATmega2560 - має цілих чотири UART.

R і T в термінології UART не несуть відповідальності за відправку та напівпродуктивних даних. На стороні передачі UART повинен створити пакет даних - додавання біт-синхронізації та можливостей - та відправити цей пакет по лінії TX відповідно до встановленої швидкості передачі. З іншого боку, UART повинен

перевірити лінійність RX із швидкістю, відповідну ожидаючу швидкість передачі в бодах, вибираючи біт-синхронізацію та викладаючи дані.

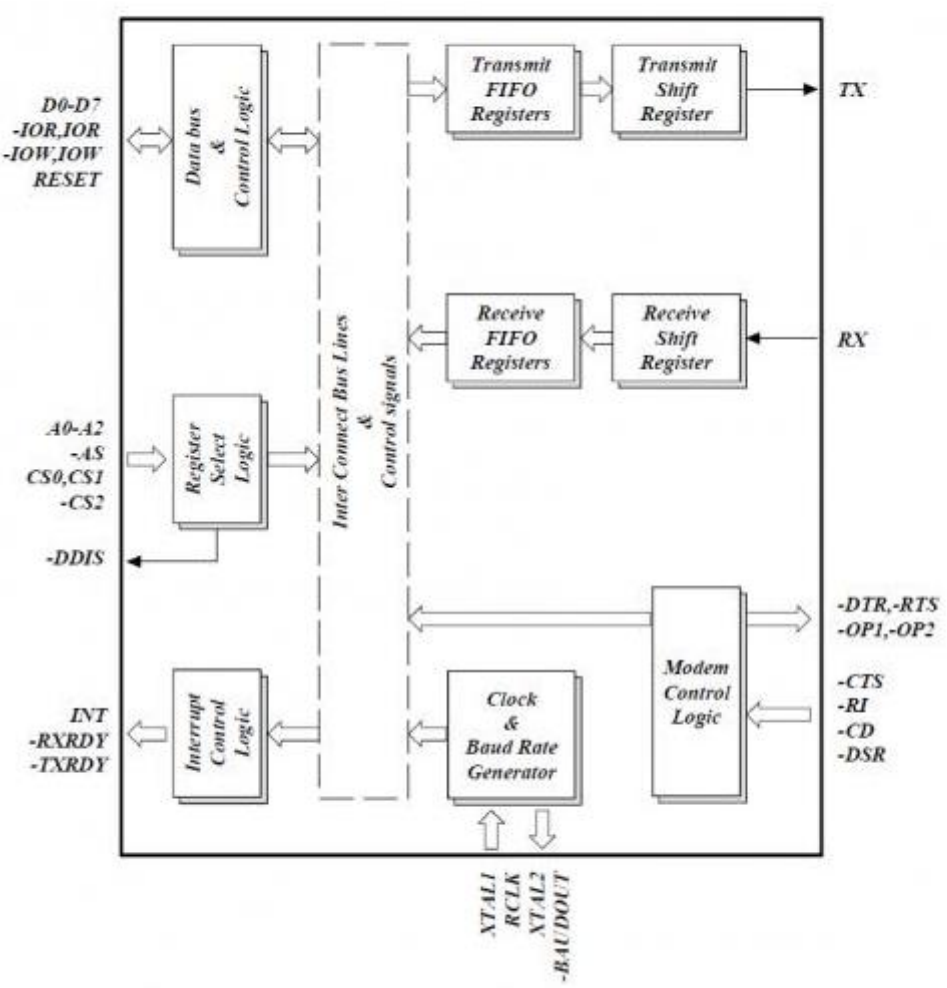

Рис. 4.2 Внутрішня блок-схема UART

Більше запропоновані UART можуть передавати отримані дані в буфер, де вони залишаться до порції, покажіть мікроконтролер, щоб не отримати їх і обробити. Зазвичай UART видає буферизовані дані за принципом "першим вошем - першим вище" (FIFO). Буфери можуть бути як маленькими, так і великими, більш тисячами байтів.

4.5 Протокол передачі даних через UART

Для UART інтерфейсу було розроблено власний протокол передачі данних, який гарантує підтвердження сторонами про отримання відправленого повідомлення. Під час розробки даного протоколу були використанні принципи роботи TCP/IP протоколів.

Оскільки через з’єднання йтиме постійний потік даних перевірки трестованої плати, дуже важливо отримувати правильні дані та уникати колізій у їх передачі. Загалом за даним протоколом, нова передача даних не відбудеться, поки відправник не отримає підтвердження отримання повідомлення такого самого розміру, якого відправник надсилав. Це дозволяє впевнитись у вірності отриманих даних та захистити систему від можливих помилок та збоїв через «бите» повідомлення.

Даний протокол має як набір важливих правил (біти про початок повідомлення, про його кінець, його довжину і т.д.), так і набір бітів-команд. Розглянемо будову коду. Нижче наведено формат структури протоколу для повідомлення:

```
protected int marker;
    protected byte messageId;
    protected byte commandId;
    protected byte argc;
    protected byte[] argv;
    protected byte crc8;
```

У вище заданому коді кожна змінні має своє значення:

— marker – індикатор початку повідомлення, який дозволяє відслідковувати нові повідомлення;
— messageId – ідентифікаційний код повідомлення;
— commandId – код команди;

— argc – кількість аргументів, які передаються;

— argv – список переданих аргументів;

— crc8 – хеш сума для перевірки правильності отриманого повідомлення.

Якщо надіслане повідомлення не має усіх зазначених бітів, або інформація про отримане повідомлення не співпадає з відправленим, повідомлення не приймається системою і вона чекає на повторення повідомлення.

Також було розроблено список команд, які потрібні для функціонування системи та для кожної було призначено власне бітове значення. Список команд виглядає наступним чином:

```
byte ACK_COMMAND_ID = (byte) 0xAB;
   byte HANDSHAKE_COMMAND_ID = (byte) 0xAC;
   byte PING_COMMAND_ID = (byte) 0xAD;

   byte PERFORM_TEST_COMMAND_ID = (byte) 0x11;
   byte CHECK_SHIELD_COMMAND_ID = (byte) 0x12;

   byte PIN_STATUS_COMMAND_ID = (byte) 0x21;
   byte SHIELD_CONNECTED_COMMAND_ID = (byte) 0x22;
   byte SHIELD_DISCONNECTED_COMMAND_ID = (byte) 0x23;
```

ACK_COMMAND_ID команда підтвердження про отримання повідомлення. Ця команда відправляється відповіддю на будь-яке повідомлення з іншою командою та таким чином повідомляє іншу сторону про успішне отримання повідомлення.

HANDSHAKE_COMMAND_ID – команда підключення. Вона застосовується для визначення сумісних пристроїв за характером відповіді на цю команду. Тобто за допомогою цієї команди система визначає пристрій та можливість його тестування.

PING_COMMAND_ID – команда, яка дозволяє перевірити конкретну ніжку тестованого апаратного засобу. Вона вимагає відправки двох параметрів: номер ніжки (1 байт) та маска статусу піна (1 байт)

PERFORM_TEST_COMMAND_ID – команда для запуску загального тестування плати. Вона запускає існуючий скрипт, який тестує усі підключені ніжки плати на працездатність.

CHECK_SHIELD_COMMAND_ID – команда, яка перевіряє підключення щита або надбудови до Arduino. Ця команда тригерить відповідну відповідь – SHIELD_CONNECTED_COMMAND_ID або SHIELD_DISCONNECTED_COMMAND_ID.

4.6 TCP/IP-з’єднання

TCP / IP - це скорочене слово для Інтернет-протоколу / протоколу. У термінології комп'ютерної мережі протокол є заздалегідь визначеним стандартом, який дозволяє двом комп'ютерам обмінюватися даними. Насправді TCP / IP - це не протокол, а декілька. Ось чому ви часто чуєте це під назвою набір або набір протоколів, де TCP та IP - це два найважливіші.

TCP / IP на вашому комп'ютері - це унікальна точка доступу для TCP, IP та інших членів сім'ї TCP / IP. Зазвичай він також має вдосконалені додатки, такі як FTP (протокол передачі файлів, протокол передачі файлів), що дозволяє контролювати обмін мережевими файлами за допомогою командного рядка.

TCP / IP - виник у дослідженнях, що фінансувалися Американським агентством прогресивних дослідницьких проектів (ARPA) у 1970-х роках. Цей протокол був розроблений для інтеграції мереж передачі даних у науково-дослідних центрах по всьому світу в "комп'ютерну мережу" (робочий процес в Інтернеті). Оригінальний Інтернет був створений заміною існуючої мережі ARPAnet на TCP / IP.

Тому що TCP / IP важливий сьогодні, оскільки він дозволяє незалежним мережам спілкуватися в Інтернеті або об'єднати зусилля, створюючи веб-сайти.

Комп'ютерні мережі, що складають фізичну інтранет, підключені до маршрутизатора або IP-роутера. Маршрутизатор - це комп'ютер, який передає пакети даних з однієї мережі в іншу. У мережі, що базується на TCP / IP, інформація передається цікавим структурам, званим IP-пакетами або IP-даними. Завдяки програмному забезпеченню TCP / IP всі комп'ютери підключені до "відносно тісної" комп'ютерної мережі. В якості основи він приховує маршрутизатори та структуру мережі та робить їх схожими на одну велику мережу. Так само як Ethernet-з'єднання ідентифікуються за допомогою 48-розрядних ідентифікаторів, внутрішньомережеві з'єднання ідентифікуються за допомогою 32-розрядних IP-адрес, які ми описали як десяткові числа, розділені на періоди (наприклад, 128.10.2.3). Передаючи IP-адресу зовнішнього комп’ютера, комп'ютер в Інтернеті чи Інтернеті може надсилати дані так, ніби вони є частиною тієї ж фізичної мережі.

TCP / IP вирішує проблеми з даними між двома комп'ютерами, підключеними до однієї інтрамережі, але належать до різних фізичних мереж. Рішення складається з декількох компонентів, кожен із сімейства протоколів TCP / IP сприяє загальним причинам. IP - базовий протокол з пакета TCP / IP - передає інформацію про Інтернет через Інтернет і виконує важливу роботу з ревом, насправді це вибір для переміщення даних з точки A в точку B та використання маршрутизатора для пропуску мереж.

TCP - протокол високого рівня, який дозволяє програмам, що працюють на різних мережевих хостах, обмінюватися потоками даних. TCP розподіляє потік даних по ланцюгах, які називаються компонентами TCP, і передає їх через IP. У більшості випадків кожен компонент TCP передається окремим IP-даним. Однак, якщо необхідно, TCP буде розподіляти частини декількох даних IP, які інтегровані у фізичний рівень шарів даних, який використовується для передачі даних між мережевими комп'ютерами. Оскільки IP не вимагає доступу до бази даних так само, як це було надіслано, TCP збирає компоненти TCP на іншому кінці шляху для отримання послідовних даних. FTP і Talent - два приклади популярних програм TCP / IP, які покладаються на TCP.

Ще один важливий член пакету TCP / IP - це протокол User Datagram (UDP), схожий на TCP, але більш досконалий. TCP - це "надійний" протокол, оскільки він

забезпечує повідомлення про перевірку помилок та перевірку помилок, щоб гарантувати, що дані досягнуть місця призначення без шкоди. UDP - це "ненадійний протокол", оскільки він не гарантує, що збір даних надійде відправленим способом, або що всі вони надійдуть. Якщо можлива ситуація слабшає, програмне забезпечення потрібно буде впровадити. Однак UDP все ще займає глобальний статус TCP / IP і використовується у багатьох програмах. Протокол протоколу управління мережею (SNMP), який реалізований у багатьох програмах TCP / IP, є лише одним із прикладів програм UDP.

Інші протоколи TCP / IP відіграють невелику роль, але також відіграють важливу роль у мережах TCP / IP. Наприклад, ARP (Address Resolution Protocol) переводить IP-адреси у фізичні мережеві адреси, такі як Ethernet. Пов'язаний протокол - RARP (протокол надійності рішення) - виконує середню дію шляхом перетворення фізичних мережевих адрес в IP-адреси. Протокол Internet Protocol (ICMP) - це протокол, який використовує IP для обміну інформацією управління та керування помилками передачі IP-пакетів. Наприклад, якщо маршрутизатор не в змозі надіслати IP-колектор, він використовує ICMP для сповіщення відправника про проблему. Короткий опис деяких інших протоколів, які можуть "приховати парасольку" TCP / IP, подано у власному контексті.

4.7 Мови програмування C та Java

Було вирішено використовувати мову C++ для програмування головного контролера, оскільки платформа Arduino має багато готових рішень та бібліотек.

Мова програмування C++ досить універсальна і відноситься до категорії високого рівня. Вона поєднує об'єктно-орієнтовані, процедурні та інші програмні парадигми.

Мова програмування C++ була створена в 1979 році датським програмістом Bjarn Straustrup. Мова отримала свою сьогоднішню назву лише через 14 років. Їй одразу дали досить логічну назву «C з класами».

C++ користувався величезною популярністю у 1990-х. Вони використовували його для різних завдань. Наприклад, розробка програмного забезпечення, відеоігор або серверних додатків. Варто зазначити, що C+ також сприяв створенню таких популярних мов програмування, як Java та C #.

Рік Массітті мав ідею змінити назву на "C ++". Він походить від звичайного оператора мови C, який був позначений символом "++". Популярним способом назви нової версії будь-якої програми було додавання префіксу "+" до імені.

Ця мова була стандартизована в 1998 році. Це зробила Міжнародна організація зі стандартизації. Так, того року стандарт C++ можна було розділити на дві частини - стандартну бібліотеку та мовне ядро. Перший також включав бібліотеку пропозицій STL, яка була розроблена одночасно зі стандартом.

Сьогодні ця бібліотека не використовується в тому ж сенсі, як раніше. Однак розробники все ще використовують примітку STL. Вони називають ту частину бібліотеки, яка зберігає такі поняття, як автори, функціонери, і містить визначення шаблонів контейнерів.

Цікаво, що мовний стандарт C++ містить посилання на мовний стандарт 1990 р. Функції, отримані з мовної бібліотеки C, не визначені в мовному стандарті C++.

Насправді, багато інших бібліотек C++ не вказані у стандарті. Таким чином, ми можемо використовувати більшу кількість мовних мов C на C++.

Ми припускаємо, що стандартизація чітко визначила мову C++. Насправді можуть бути елементи інших неповних мов програмування, які не є стандартизованими, оскільки ця мова спочатку існувала окремо і розроблялася тоді, коли потрібно було виконати нове завдання. Одночасно готувався компілятор під назвою Cfront.

Мови програмування на C++, як і інші, мають стандартну бібліотеку. Він складається із стандартної бібліотеки C, але є відмінності, необхідні для кращої роботи з іншою мовою. Але мовна бібліотека C++ також має свою частку на основі STL. Існують дуже потрібні інструменти, такі як ітератори та контейнери, які містять такі поняття, як списки, вектори тощо. Їх можна використовувати для доступу до

контейнерів та матриць. Використовуючи STL, можна аналогічно працювати з контейнерами зовсім іншого типу: черги, списки та інші.

Без шаблонів неможливо використовувати узагальнені алгоритми, які могли б працювати з суднами або навіть послідовностями, де ітератори дозволяють отримати прямий доступ до членів.

Використання функцій бібліотеки мови C++ реалізовано за допомогою директиви #include. Ця дія активує стандартні файли, у тому числі 50 штук.

STL спочатку був проектом HP. Пізніше вона була розроблена SGI. А потім він був включений у мовний стандарт C++, але без використання назви "STL". Тим не менш, велика кількість програмістів все ще називають це так, що воно відрізняється від стандартної бібліотеки простотою. Спеціальним проектом STLport є оновлення STL. Однак він не єдиний. Є й інші проекти, які також розроблять інші програми STL для різних завдань. Цікаво, що при створенні компілятора C++ повинна бути надана бібліотека. Адже вона стала невід’ємною частиною стандарту і завжди активно використовується.

Програмування веб-інтерфейсу та основна бізнес-логіка були розроблені на Java за допомогою інструментів Vaadin.

Java - об'єктно-орієнтована мова програмування, опублікована Sun Microsystems у 1995 році як центральний компонент платформи Java. Oracle працює з мовою з 2009 року і придбав Sun Microsystems в тому ж році. В офіційній версії програми Java складаються в байт-код, який інтерпретується віртуальною машиною для певної платформи.

Мова запозичила багато синтаксису з C і C++. Об'єктом є в основному модель об'єкта C++, яка була змінена. Можливість виникнення деяких конфліктних ситуацій, які можуть виникнути через програмні помилки, усунена, а процес розробки об'єктно-орієнтованих програм полегшено. Віртуальній машині призначається кілька дій, які оператори повинні виконувати в C / C++. Перш за все, Java була розроблена як незалежна від платформи мова, тому вона має менше апаратних функцій низького рівня, що знижує швидкість роботи додатків порівняно, наприклад, з C++. Java дозволяє викликати підпрограми, написані іншими мовами.

4.8 Обгрунтування вибору програмної реалізації

Розробляючи систему та проаналізувавши предметну область та вимоги замовника було вирішено розробити програмний продукт на основі веб-технологій для використання з веб-браузером. Фреймфорк Vaadin має зручні і легкі у використанні інструменти для створення веб-інтерфейсу та взаємодії з сервером. Java технології, які використовуються на сервері, були вибрані на основі принципу зручності використання, відкритої системи, актуальності та можливості роботи в будь-якій операційній системі. Зокрема набору бібліотек для роботи з апаратними можливостями платформи Orange Pi

Для побудови фізичної складової апаратно-програмної частини було обрано Orange Pi, Arduino Mega та каскаду мультиплексорів hc4067. При виборі апаратного забезпечення важливо було поєднати надійність та ефективність використання, стабільністю роботи плат з максимальною тривалістю використання, а також низькою ціною.

Orange Pi - одна з найвідоміших брендів серед планшетних комп'ютерів. Вартість Orange Pi починається від 15 доларів. Це одна з найдешевших одиниць з хорошими можливостями [3]. Переваги Orange Pi:

— Низька ціна - плати Orange Pi приблизно в 2 рази дешевші, ніж Raspberry Pi;
— Різноманітні моделі, придатні для спеціальних завдань;
— Сумісність GPIO;
— Сумісність з іншими компонентами для Orange комп'ютерів;
— Висока швидкість;
— Великий об'єм пам'яті;
— Наявність декількох роз'ємів USB (залежно від моделі) та виходів HDMI;

— Велика кількість операційних систем.

Модуль Arduino MEGA 2560 набагато більший за інші (101,52 × 53,3 мм). Він заснований на мікроконтролерах ATmega2560 і має значно більше пам’яті. Кілька портів - 54 цифрових вводу / виводу (з них 15) можна використовувати для ШІМ), 16 аналогових входів. Насправді, Arduino Mega 2560 - це розширена версія Arduino Uno. Він розроблений саме так, щоб бути сумісним з "маленькими братами" та модулями розширення (щитами). Таким чином, ліва сторона дошки така ж, як і у Arduino Uno, з точки зору як макета, так і макета штифтів. Завдяки цьому, Arduino Mega замінить плату Arduino Uno, якщо її недостатньо.

Arduino Mega 2560 відрізняється від усіх попередніх плат тим, що використовує мікроконтролер ATmega16U2 (ATmega8U2 у версіях R1 та R2 карти) замість мікросхеми FTD для перетворення інтерфейсу USB-UART.

До картки R2 Mega 2560 версії додано резистор, який малює на землі лінію HUB мікроконтролера 8U2. Такий захід спрощує процес оновлення мікропрограмного забезпечення та переключення пристрою в режим DFU.

Pinout 1.0: Додано цвяхи SDA та SCL (біля штифта AREF), а також два нові штифти, розташовані біля шпильки RESET. Перший - IOREF - дозволяє розширювальним платам адаптуватися до робочої напруги Arduino. Цей висновок спрямований на сумісність карт розширення як з 5 V Arduino на базі мікроконтролерів AVR, так і з 3.3V платами Arduino Due. Другий вихід не пов'язаний ні з чим і зарезервований для майбутніх цілей. Підвищена шумостійкість для скидання ланцюга. Мікроконтролер ATmega16U2 замінено на 8U2.

5 ІНТЕРФЕЙС КОРИСТУВАЧА

Розроблений інтерфейс апаратно-програмного комплексу призначений для мобільних платформ та пристроїв IOS та Android.

5.1 Інсталяція та системні вимоги

Щоб встановити додаток, потрібно завантажити його на свій пристрій та встановити відповідно до методів операційної системи. Стабільна передача даних також вимагає стабільного доступу до Інтернету зі швидкістю не менше 5 Мбіт / с. Щоб використовувати можливості програми віддаленого управління платформою, ви повинні надати їй доступ до роботи з мережею.

5.2 Інструкція з використання програмного продукту

Для користування програмним застосунком користувача, користувач повинен відкрити у актуальній версії веб-браузеру Chromium відкрити адресу серверу.

Рис. 5.1 Головне меню сайту

Використовуючи програмний застосунок, користувач може виконувати тестування різних тестових програм на різних цільових апаратних платформах. Для цього, користувачу необхідно обрати бажані параметри у відповідному меню. Після цього, користувач може запустити тестування.

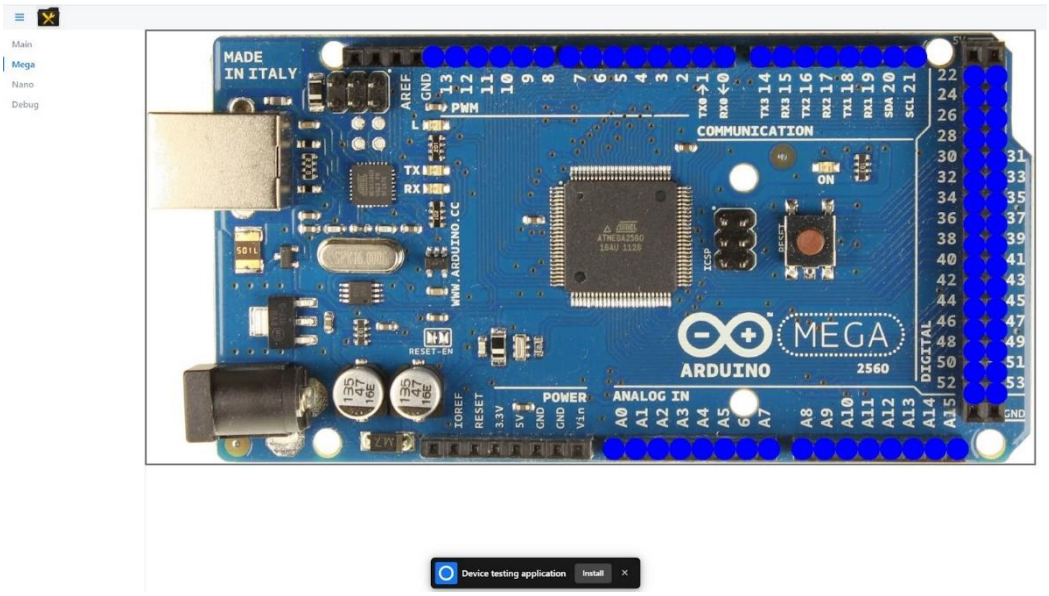

Рисунок 5.2.1 Зображення інтерфейсу стану роботи Arduino Mega

У поточній версії програми, користувачу доступні апаратні платформи Arduino Mega (рисунок 5.2.1) та Arduino Nano (рисунок 5.2.2), з відповідними інтерфейсами стану роботи.

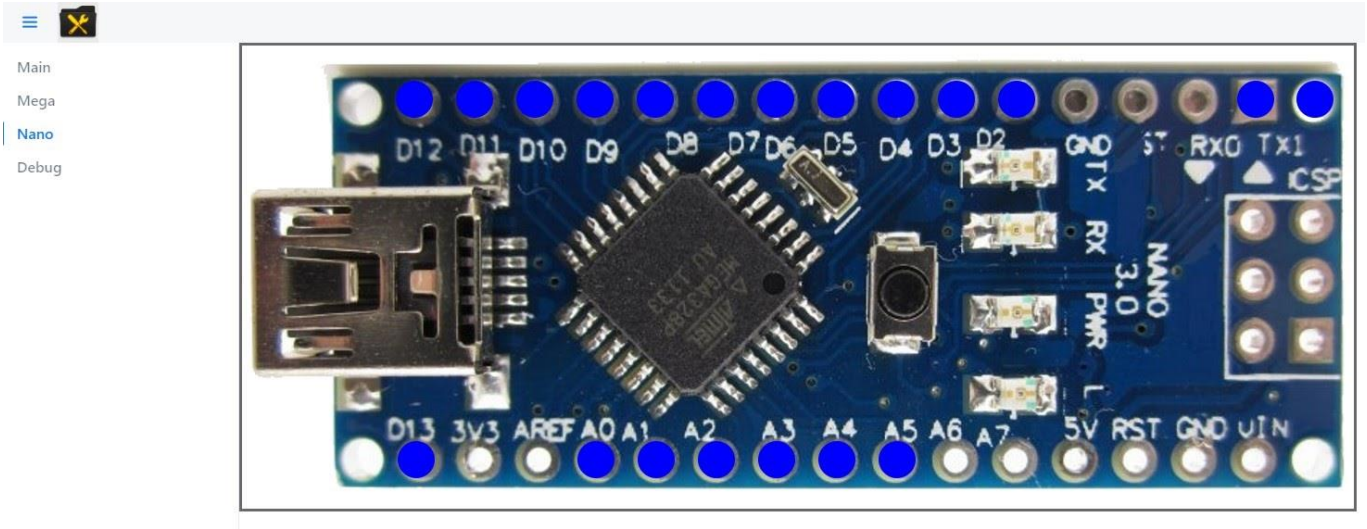

Рисунок 5.2.2 Зображення інтерфейсу стану роботи Arduino Nano

У відповідному апаратній платформі інтерфейсі можна у реальному часі спостерігати за процесом тестування. За результатами тестування, користувач отримає детальний опис у відповідному меню.

ВИСНОВКИ

У ході аналізу тестування апаратного забезпечення було досліджено існуючі методи тестування апаратного забезпечення. Аналіз показав, що не існує систем, які вирішують задачу у повному обсязі, а також не існує автоматизованих способів тестування

Розроблений програмний продукт дозволяє автоматизувати тестування апаратного забезпечення та швидко визначати межі можливостей різноманітних платформ.

Проведено огляд методів і засобів проектування та розробки апаратно-програмної системи. Обґрунтовано вибір архітектури апаратної складової, а також програмної архітектури, яка базується на веб-технологіях. Це дає змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використані.

За результатами виконання тестових завдань підтверджена коректність отриманих результатів, отже система відповідає поставленим вимогам.

Користувачами системи можуть бути різноманітні користувачі, як досвідчені розробники апаратно-програмного забезпечення, так і початківці, які лише починають знайомитись з апаратними платформами. Апаратно-програмний комплекс може бути використано на будь-якій операційній системі, на якій встановлено браузер, який підтримує останні веб-стандарти, а також яка має постійний доступ до інтернету.

Отже, практика покращила знання різноманітних технологій, що використовуються під час розробки проектування та розробки апаратно-програмних комплексів. Також було досліджено різноманітні техніки та методи тестування можливостей апаратних платформ, детально досліджено тонкощі у використанні різних типів платформ.

Було розглянуто принципи проектування та побудови власних апаратних платформ, а також правила екологічно стійкого дизайну: шляхи подовження використання плат та контролерів, проектування апаратних платформ з максимально

тривалим життєвим циклом, які після використання піддаються утилізації або переродженню, тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Autodesk Design Academy — Improving Product Lifetime: Repair & Upgrade — 2020.
2. The Elements of Sustainable Design in the Consumer Electronics Industry — 2019 — [Електронний ресурс]. Режим доступу: https://www.ul.com/news/elements-sustainable-design-consumer-electronics-industry.
3. Orange Pi конкурент чи наслідувач Raspbery Pi? – [Електронний ресурс]. Режим доступу: https://arduinomaster.ru/raspberry-pi/orange-pi-konkurent-ili-podrazhatel-dlya-raspberry-pi/
4. Barnes, D., & Kölling, M. (2017). Objects first with Java. Harlow: Pearson.
5. Arndt, D. (2014). Vaadin. [Place of publication not identified]: Dpunkt.
6. Martin, R. (2019) Clean Architecture.
7. Hayes, T., & Horowitz, P. (2010). Student manual for the art of electronics. Cambridge: Cambridge University Press.
8. 5 Reasons Why Enterprises Use Vaadin For Their Web Application UI Needs. (2020). - [Електронний ресурс]. Режим доступу: https://medium.com/@ankurmans/an-open-letter-to-all-enterprise-level-business-web-application-decision-makers-446527292627
9. ATMEGA2560 Datasheet, PDF - Alldatasheet. (2020). - [Електронний ресурс]. Режим доступу: https://www.alldatasheet.com/view.jsp?Searchword=Atmega2560&gclid=Cj0KCQjw-_j1BRDkARIsAJcfmTEe7p3PeFou_2jKv2UzAGMuHIArDgUbxmESsEWOsuBhr4915hdtqVcaAoMuEALw_wcB
10. Arduino Mega [Електронний ресурс]. Режим доступу: https://doc.arduino.ua/ru/hardware/Mega2560
11. Orange Pi [Електронний ресурс]. Режим доступу: http://www.orangepi.org/

12. Схемотехніка: Мультиплексори [Електронний ресурс]. Режим доступу: http://mcx.lab-101.org.ua/Tema17.htm
13. Vaadin traning [Електронний ресурс]. Режим доступу: https://vaadin.com/learn/training/v14
14. Нарозування розмірності мультиплексорів [Електронний ресурс]. Режим доступу: https://studfile.net/preview/5199065/page:3/

ДОДАТОК 1

Апаратно-програмний комплекс тестування технічного стану апаратних платформ.

Специфікація

КР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ6174_20Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ6174_20Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ6174_20Б 12-1	DBConnectionEvent.java DBUpdateService.java DoorService.java EventService.java GpioInputListener.java GPIO.java ModelController.java	Основні компоненти
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ6174_20Б 12-2	Додаток 2.doc	Опис програмного модуля

ДОДАТОК 2

Апаратно-програмний комплекс тестування технічного стану апаратних платформ.

Лістинг програмного модулю

КР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ6174_20Б

Аркушів 7

Київ 2020

```
package com.denvys5.controller.service;

import com.denvys5.Settings;
import com.denvys5.model.Entity.Event;

import java.util.Date;

public class DBConnectionEvent extends Thread {
    private EventService eventService = EventService.getInstance();
    public void run(){
              eventService.addEvent(new Event(Settings.doorName, "System
has started", new Date()));
        while (!isInterrupted()){
            try {
                Thread.sleep(3600000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            eventService.addEvent(new Event(Settings.doorName, "System is
running as usual", new Date()));
        }
    }
}
package com.denvys5.controller.service;

import com.denvys5.Settings;
import com.denvys5.controller.service.socket.SocketEventListeners;
import com.denvys5.model.DAO;
import com.denvys5.model.Entity.Door;
import com.denvys5.model.Entity.Event;
import com.denvys5.model.dto.DoorDTO;
import io.netty.channel.ChannelHandlerContext;

import java.util.Date;

public class DBUpdateService {
    private static DBUpdateService ourInstance = new DBUpdateService();

    public static DBUpdateService getInstance() {
        return ourInstance;
    }

    private DBUpdateService() {
        eventService = EventService.getInstance();
        socketEventListeners = SocketEventListeners.getInstance();
        socketEventListeners.addReceiveMessageEventListener((args) ->
onDBRecieved((ChannelHandlerContext) args[0], args[1]));
    }
    private SocketEventListeners socketEventListeners;
    private EventService eventService;


    public void onDBRecieved(ChannelHandlerContext ctx, Object message){
        if(message instanceof DoorDTO){
            DoorDTO doorDTO = (DoorDTO) message;
            if(doorDTO.name.equals(Settings.doorName)){
                DAO.getInstance().wipeAndUpdateDB(new Door(doorDTO));
                eventService.addEvent(new Event(Settings.doorName, "DB has
been updated", new Date()));
            }else{
```

```
                eventService.addEvent(new Event(Settings.doorName, "Invalid
DB update", new Date()));
            }
        }
    }
}
package com.denvys5.controller.service;

import com.denvys5.Settings;
import com.denvys5.model.DAO;
import com.denvys5.model.Entity.Event;
import com.denvys5.model.Entity.NFC;
import com.denvys5.model.Entity.User;
import com.denvys5.model.dao.DoorDAO;
import com.denvys5.model.dao.NFCDAO;

import javax.persistence.NoResultException;
import javax.persistence.Query;
import java.util.Date;
import java.util.List;

public class DoorService {
    private static DoorService ourInstance = new DoorService();
    private DAO dao;
    private DoorDAO doorDAO;
    private NFCDAO nfcdao;
    private EventService eventService;
    private DBUpdateService dbUpdateService;

    public static DoorService getInstance() {
        return ourInstance;
    }

    private DoorService() {
        dao = DAO.getInstance();
        doorDAO = DoorDAO.getInstance();
        nfcdao = NFCDAO.getInstance();
        eventService = EventService.getInstance();
        eventService.start();
        dbUpdateService = DBUpdateService.getInstance();
    }


    public User getUserFromUID(String uid) throws NoResultException {
        NFC nfc = nfcdao.getNFCFromUID(uid);
        return nfc.getUser();
    }

    public void onDoorOpen(String uid){
        User user = getUserFromUID(uid);
        StringBuilder message = new StringBuilder();
        message.append("Door ")
                .append(Settings.doorName)
                .append(" is opened.");
        eventService.addEvent(new Event(user.getLogin(),
message.toString(), new Date()));
    }

    public boolean isNfcInDB(String uid) {
        Query query = dao.createQuery("from NFC where uid = :param");
```

```
        query.setParameter("param", uid);
        List<NFC> nfcs = query.getResultList();
        if(nfcs.isEmpty()) {
            eventService.addEvent(new Event(Settings.doorName, "Somebody,
with uid " + uid + " tried to access the Door", new Date()));
            return false;
        }
        return
nfcs.get(0).getUser().getDoors().contains(doorDAO.getDoorFromName(Settings.
doorName));
    }
}
package com.denvys5.controller.service;

import com.denvys5.controller.service.socket.ConnectionHolder;
import com.denvys5.model.Entity.Event;
import com.denvys5.model.dao.EventDAO;
import com.denvys5.model.dto.EventDTO;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CopyOnWriteArrayList;

public class EventService extends Thread {
    private static EventService ourInstance = new EventService();

    public static EventService getInstance() {
        return ourInstance;
    }

    private EventService() {
        eventDAO = EventDAO.getInstance();
        connectionHolder = ConnectionHolder.getInstance();
    }

    private EventDAO eventDAO;
    private ConnectionHolder connectionHolder;
    private List<EventDTO> eventsToSend = new CopyOnWriteArrayList<>();

    public void run(){
        while(!isInterrupted()){
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            if(eventsToSend.isEmpty())
                continue;

            connectionHolder.writeToChannel(new ArrayList<>(eventsToSend));
            eventsToSend.clear();
        }
    }

    public void addEvent(Event event){
        eventDAO.addEvent(event);
        eventsToSend.add(new EventDTO(event));
    }
}
package com.denvys5.controller.service;
```

```
import com.denvys5.controller.GPIO;
import com.pi4j.io.gpio.GpioPinDigitalInput;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinState;

import java.io.IOException;

public class GpioInputListener extends Thread {

    private GpioPinDigitalInput arduinoInput;
    private GpioPinDigitalOutput redDiod;

    public GpioInputListener(GPIO gpio) {
        this.arduinoInput = gpio.getArduinoInput();
        this.redDiod = gpio.getRedDiod();
    }

    @Override
    public void run() {
        while(!isInterrupted()){
            if(arduinoInput.getState().isLow()){
                redDiod.setState(PinState.HIGH);
                System.out.println("Shutdown started");
                try {
                    Runtime.getRuntime().exec("sudo /sbin/shutdown -h
now");
                } catch (IOException e) {
                    e.printStackTrace();
                }
                this.interrupt();
            }
        }
    }
}
package com.denvys5.controller;
import com.denvys5.controller.service.GpioInputListener;
import com.pi4j.io.gpio.*;
import com.pi4j.io.gpio.event.GpioPinDigitalStateChangeEvent;
import com.pi4j.io.gpio.event.GpioPinListenerDigital;
import com.pi4j.util.CommandArgumentParser;

import java.io.IOException;

public class GPIO extends Thread{
    private static GPIO ourInstance = new GPIO();

    public static GPIO getInstance() {
        return ourInstance;
    }

    private GPIO() {
    }
    private GpioPinDigitalOutput outputLock;
    private GpioPinDigitalOutput arduinoOut;
    private GpioPinDigitalOutput redDiod;
    private GpioPinDigitalOutput yellowDiod;
    private GpioPinDigitalOutput greenDiod;
    private GpioPinDigitalInput arduinoInput;
    private GpioInputListener arduinoInputListener;
```

```
    private GpioController gpio;
    public void run(){
        setUp();
    }

    public void setUp(){
        gpio = GpioFactory.getInstance();
//gerkon gpio31
        System.out.println("We are initializing gpio");

        Pin redDiodPin = CommandArgumentParser.getPin(OrangePiPin.class,
OrangePiPin.GPIO_22);
        redDiod = gpio.provisionDigitalOutputPin(redDiodPin, PinState.LOW);
        redDiod.setShutdownOptions(true, PinState.LOW);

        Pin yellowDiodPin = CommandArgumentParser.getPin(OrangePiPin.class,
OrangePiPin.GPIO_23);
        yellowDiod = gpio.provisionDigitalOutputPin(yellowDiodPin,
PinState.LOW);
        yellowDiod.setShutdownOptions(true, PinState.LOW);

        Pin greenDiodPin = CommandArgumentParser.getPin(OrangePiPin.class,
OrangePiPin.GPIO_24);
        greenDiod = gpio.provisionDigitalOutputPin(greenDiodPin,
PinState.LOW);
        greenDiod.setShutdownOptions(true, PinState.LOW);

        Pin pin = CommandArgumentParser.getPin(OrangePiPin.class,
OrangePiPin.GPIO_26);
//        Pin pin = CommandArgumentParser.getPin(OrangePiPin.class,
OrangePiPin.GPIO_01);
        outputLock = gpio.provisionDigitalOutputPin(pin, "My Output",
PinState.LOW);
        outputLock.setShutdownOptions(false, PinState.LOW);

        Pin ipin = CommandArgumentParser.getPin(OrangePiPin.class,
OrangePiPin.GPIO_12);
        PinPullResistance pull =
CommandArgumentParser.getPinPullResistance(PinPullResistance.PULL_DOWN);
        arduinoInput = gpio.provisionDigitalInputPin(ipin, pull);
        arduinoInputListener = new GpioInputListener(this);
        arduinoInputListener.start();
        arduinoInput.setShutdownOptions(true);

        System.out.println("We have initialized gpio");
        Pin tinypin = CommandArgumentParser.getPin(OrangePiPin.class,
OrangePiPin.GPIO_11);
        arduinoOut = gpio.provisionDigitalOutputPin(tinypin, PinState.LOW);
        arduinoOut.setShutdownOptions(true, PinState.LOW);
        arduinoOut.setState(PinState.HIGH);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        arduinoOut.setState(PinState.LOW);

        greenDiod.setState(PinState.HIGH);
        yellowDiod.setState(PinState.HIGH);
        redDiod.setState(PinState.LOW);
```

```
    }

    public void openDoor(){
        outputLock.pulse(400, true);// set second argument to 'true' use a
blocking call
    }

    public GpioPinDigitalOutput getRedDiod() {
        return redDiod;
    }

    public GpioPinDigitalOutput getYellowDiod() {
        return yellowDiod;
    }

    public GpioPinDigitalOutput getGreenDiod() {
        return greenDiod;
    }

    public GpioPinDigitalInput getArduinoInput() {
        return arduinoInput;
    }

    public void shutdown(){
        gpio.shutdown();
        this.interrupt();
    }

}
package com.denvys5.controller;


import com.denvys5.controller.service.DBConnectionEvent;
import com.denvys5.model.DAO;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class ModelController extends Thread{
    private static ModelController ourInstance = new ModelController();

    public static ModelController getInstance() {
        return ourInstance;
    }

    private ModelController() {
    }
    private EntityManagerFactory sessionFactory;
    private EntityManager entityManager;
    private boolean ready = false;

    private void setUp() {
        boolean connectionEstablished = false;
        while(!connectionEstablished){
            try{
                sessionFactory = Persistence.createEntityManagerFactory(
"DB" );
```

```
                connectionEstablished = true;
            }catch (Exception e){
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e1) {
                    e1.printStackTrace();
                }
                e.printStackTrace();
                System.err.println("No connection to DB");
            }
        }
        System.out.println("Connection established");
        entityManager = sessionFactory.createEntityManager();
        DAO.setUpDependencyInjection(entityManager);
    }

    public boolean isReady(){
        return ready;
    }

    public void shutdown(){
        entityManager.close();
        sessionFactory.close();
    }

    public void run(){
        setUp();
        DBConnectionEvent dbConnectionEvent = new DBConnectionEvent();
        dbConnectionEvent.start();
        System.out.println("Model is fully initialized!");
        ready = true;
    }
}
```

ДОДАТОК З

Апаратно-програмний комплекс тестування технічного стану апаратних платформ.

Опис програмного модулю

КР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ6174_20Б

Аркушів 8

Київ 2020

Анотація

Розділ містить опис програмної частини, яка слугує для забезпечення прийому та передачі данних. Саме цей модуль займається обробкою та генерацією повідомлень.

ЗМІСТ

Загальні відомості

У додатку міститься частина коду, яка забезпечує виконання функцій тестування апаратних пінів.

Програма була написана мовою Java у середовищі розробки JetBrains Intellij IDEA.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

У даній частині програми реалізуються такі функції:

1. Прийом даних контролера
2. Обробка даних контролера
3. Прийняття рішення щодо статусу піна
4. Виклик відповідних апаратних інтерфейсів для тестування

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Модуль реалізований у формі набору класів. Він забезпечує зв’язок між користувачем і модулем тестування, отримуючи дані, реагуючи на них та відправляючи відповідні команди у реальному часі. Тобто реалізує виконання тестів.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модуль розроблено у середовищі розробки JetBrains Intellij IDEA. При роботі програми використовуються такий пристрій як апаратна платформа модулю забезпечення доступу до пінів, що складається з таких пристроїв як мікрокомп’ютер OrangePi PC, мікроконтролер Atmega 328p, апаратна платформа Arduino Nano, та периферійних модулів.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Система забезпечує постійний контроль роботи в реальному часі, тому цей модуль запускається автоматично після запуску програмного забезпечення відповідного агенту і забезпечує виконання тестування.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є інформація, яку надсилає користувач при авторизації. Це може запит, що містить у собі дані про контроллер користувача.

Вихідними ж є результат тестування (наприклад, відкривання дверей) у разі якщо цього дозволяє регламент доступу.